On Optimal Strategies for Wordle

Alexander D. Healy

January 9, 2022 (Revised January 18, 2022)

Abstract

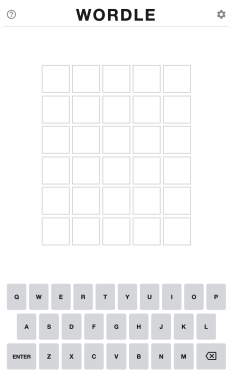
This note describes the methodology behind the (near-optimal) Wordle strategy available at: `http://www.alexhealy.net/papers/wordle/wordle-TRACE.html`, as well as other similar strategies.

Introduction

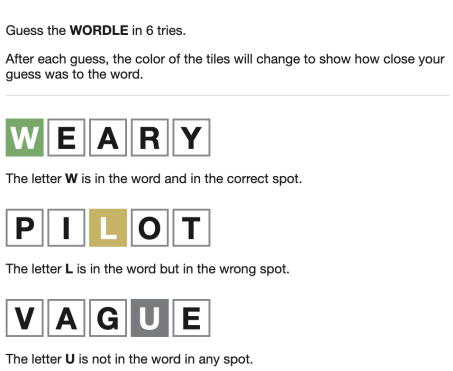
Wordle[1] is a word-guessing game that has enjoyed a rapid rise in popularity recently (early 2022). This note describes some strategies for playing Wordle, based on some basic ideas from *information theory*. While, many of the ideas/claims presented here can be formalized rigorously, this presentation is intentionally somewhat more informal.

Game Play

The goal of Wordle is to guess a hidden 5-letter word, using up to 6 guesses. The guesses are entered (one at a time) in a grid, and the instructions on the website succinctly describe the mechanics of the game:



(a) Empty Wordle grid



(b) How to play Wordle

Optimal Play

Part of the fun of games like Wordle is the intellectual challenge of finding the solution using human intelligence and intuition. This note completely undermines this appeal by asking how we can, in a purely mechanical fashion, play Wordle "optimally".

Several authors have written on similar themes. For example, Matt Rickard's blog post "What's the Best Starting Word": `https://matt-rickard.com/wordle-whats-the-best-starting-word/`
Bertrand Fan's blog post "The Best Starting Word in Wordle":
`https://bert.org/2021/11/24/the-best-starting-word-in-wordle/`

[1]The official Wordle site is at: `https://www.powerlanguage.co.uk/wordle/`

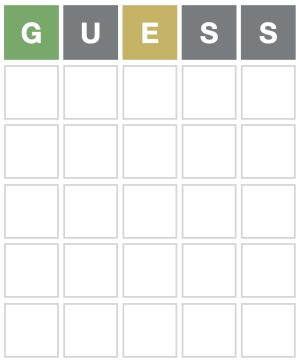
as well as Leonardo Taccari's blog post "Solving wordle":
`https://leotac.github.io/posts/2022/01/13/wordle/`
among others.

Of course, the notion of an "optimal" or "best" strategy depends on our objectives. Is the goal simply to never lose? Or to win with the fewest number of guesses on average? Or to win with at most 4 guesses as often as possible? Etc. The approach outlined below offers some strategies that are very competitive along these various dimensions.

The Entropy Approach

One goal of this note is to exploit a natural connection between Wordle puzzles and the notion of *entropy* in information theory.[2] Informally, "entropy" is a measure of the "uncertainty" in some random event. For example, flipping a (fair) coin represents an "entropy" of 1 because there are two equally-likely outcomes, and so there is exactly "1 bit" of uncertainty about the outcome. Similarly, a look at the source code for Wordle reveals that there are 2315 valid solution words; so, if we assume the solution to any given puzzle is chosen randomly, then the entropy associated with the unknown solution is $\log_2 2315 = 11.176796\ldots$ bits of information. The goal of the game is to determine, with absolute certainty, what the hidden word is. In other words, we want to drive this entropy down to 0 (i.e., no uncertainty), and this is possible because each guess reveals information about the solution, narrowing-down the set of possible solutions and hence driving down the entropy, until there is only one viable solution remaining.

To illustrate this process, suppose we make a guess ("GUESS"):



The colors (green, yellow, black[3]) now reveal some information about the hidden word, and the goal of our guesses will be to choose words that makes this information as useful as possible. The natural way to measure the usefulness of a guess is to consider the *expected entropy* of a randomly chosen solution, conditioned on the information learned from the guess. Intuitively, this asking: how much entropy/uncertainty do I expect to be leftover (in the hidden word) once I've made my guess (and have seen the resulting pattern of green/yellow/black indicating how consistent my guess is with the solution.)? And for any given guess, this expected entropy can easily be calculated as described below.

First, observe that every pattern of green/yellow/black hints corresponds to a subset of solutions that's consistent with that hint. (In the above example, the solution GIVER is consistent with the hint, but the solution HALVE is not, for example.) So, if S denotes the set of all 2315 possible solutions, we can split that space into disjoint sets

$$S = S_{\text{GBYBB}} \cup S_{\text{GBBBG}} \cup \cdots$$

[2]For some background on entropy see, e.g., `https://en.wikipedia.org/wiki/Entropy_(information_theory)`)

[3]In the game, this color is rendered as gray, whereas it appears as black in the game summaries posted on social media. We'll simply call this gray-ish color "black" (abbreviated "B") to avoid confusion with green ("G").

based on which solutions are consistent with each pattern of G/Y/B. (For example, GIVER would be in the set S_{GBYBB}.) Now, the probability that the solution is in one of these sets is simply $|S_{\text{pattern}}|/|S|$, so the expected entropy of the unknown solution (following the feedback on our guess) is:

$$\text{Entropy}(\text{solution}|\text{guess}) = \sum_{\substack{\text{pattern} \\ \text{of G/Y/B}}} \log_2(|S_{\text{pattern}}|) \cdot \frac{|S_{\text{pattern}}|}{|S|},$$

which can easily be calculated by, say, a Python program. In the case of the guess "GUESS" shown above, this entropy works out to 7.203803..., which is much better than the initial entropy of 11.176796... — effectively we learn nearly 4 bits of information about the solution, on average, by guessing "GUESS" as our first move. But this is not as good as the optimal first guess, which turns out to be "SOARE" (reducing the entropy to just 5.290836...), and just narrowly wins out over "ROATE" (5.294017...), "RAISE" (5.298887...), etc.

It's also worth noting that the pattern "GGGGG" has a special status: it means we've won and the game ends (in addition to the conditional entropy being 0). Clearly, we prefer a guess that wins to one that simply drives the entropy to 0.[4] To deal with this in practice, it's convenient to modify the entropy calculation so that $\log_2(|S_{\text{GGGGG}}|)$ has a value of -1, rather than 0. This effectively gives a *bonus* of "1 bit" of entropy for a win. (Even after this modification SOARE is the best first guess, but on smaller sub-problems the modification becomes more useful.)

Finally, by recursively applying the same analysis to each of the sets S_{pattern}, we can construct a tree where each node says what the optimal guess is to play based on preceding guesses and their resulting hints.

Some Observations

Armed with our tree of guesses, we can answer some other basic questions about this Wordle strategy. For example: for each possible hidden word, how many steps does this strategy require? Here is that distribution:

# guesses	# words
2	46
3	1217
4	990
5	61
6	1

So, for example, all words are found in at most 6 guesses (which is fortunate, because that's the maximum number of guesses allowed!) and 46 of the words are found in just 2 guesses.
The single word that requires 6 guesses using this approach is WAVER.
The average number of guesses required (across all solutions) is 3.461771....

Going Further

Can we do better? While the above algorithm attempts to make an "optimal" guess at each stage, it is only looking one move ahead in assessing the quality of a guess. However, it's plausible that some "sub-optimal" guesses at the beginning actually yield better results later in the search. For example, if we start with a guess of "TRACE" (rather than "SOARE")[5] and then follow exactly the same procedure for the rest of the tree, we end up with a strategy that solves all Wordle puzzles in at most 5 guesses, has an average number of guesses of 3.434557... and has the following distribution of the number of guesses:

[4] For example, suppose the search is limited to two possible solutions: PICKY and PIGGY. It's possible to distinguish between these by guessing CIGAR, but it's clearly better to guess PICKY (or PIGGY), in which case there's a 50% chance of winning immediately.

[5] An initial guess of "TRACE" yields an expected conditional entropy of 5.346... (vs. 5.291... for "SOARE"), making it the 10th-best starting guess, according to this measure. This is true for both for the pure conditional entropy calculation as well as for the modified entropy calculation described above.

# guesses	# words
1	1
2	75
3	1214
4	967
5	58

In particular, this strategy solves the vast majority of Wordles in at most 4 guesses and only requires 5 guesses $58/2315 \approx 2.5\%$ of the time.

Thus far, the strategies we've considered make all their decisions by looking at the expected conditional entropy of the next guess, without explicitly looking further ahead. How much better are the guesses if we look, say, *two* moves ahead when calculating the conditional entropy?[6] This is a much heavier calculation that involves (i) iterating over all choices of an initial guess, (ii) finding the minimum-conditional-entropy guess on each resulting branch, (iii) computing the weighted sum of these optimal conditional entropies. An extensive computer search found that the optimal initial guess using this approach is "SLANE", which has an expected conditional entropy (after two guesses) of just $1.1407\ldots$ bits. One corollary of this search is that *no strategy can solve all Wordle puzzles in at most* 3 *guesses*. Indeed, any strategy that always succeeds after 3 guesses must know the solution with absolute certainty after the second guess, which would imply an expected conditional entropy (after two guesses) of 0.

This strategy of looking two steps ahead can also help improve the "TRACE" tree described above. Specifically, if we start with a guess of "TRACE" and then build the rest of the game tree using two-step conditional entropy calculations to pick the best guess at each stage, we obtain a strategy that's even better still; it has an average number of guesses of $3.430669\ldots$ and solves all but $40/2315 \approx 1.7\%$ of Wordles in 4 guesses or less:

# guesses	# words
1	1
2	75
3	1205
4	994
5	40

An interactive webpage with the entire game tree for this strategy is available at:
`http://www.alexhealy.net/papers/wordle/wordle-TRACE.html`.

One possible criticism of this strategy is that it can make use of obscure/archaic words such as DIOLS[7] and DHOWS[8] as it homes in on the solutions. To discourage this behavior, we can limit the guesses to be from the set of 2315 valid solutions (which are generally more common words). Remarkably, this yields some very competitive game trees, despite having fewer than one-fifth as many guesses to choose from. For example, starting with the root "REACT" (an anagram of "TRACE"!), this generates a strategy with an average number of guesses of $3.476026\ldots$ and the following distribution of guesses:

# guesses	# words
1	1
2	84
3	1096
4	1080
5	54

An interactive webpage with the entire game tree for this strategy is available at:
`http://www.alexhealy.net/papers/wordle/wordle-REACT-SolsOnly.html`.

[6] Thanks to Noam D. Elkies for raising this question.
[7] Alcohol molecules containing two hydroxyl groups.
[8] Two-masted Arab sailing vessels.

Other Questions

Based on the above analysis, we know that no strategy can always win in 3 guesses, and the "TRACE" strategy always wins with a maximum of 5 guesses. Is there a strategy that requires at most 4 guesses? It turns out the answer is No: Russ Cox[9] has done an extensive computation ruling out any 4-guess strategies. What is the average number of guesses needed by an optimal strategy? Is it materially less than 3.43?

Acknowledgements

Thanks to Dave Fetterman, Noam Elkies and, in particular, Russ Cox for various thought-provoking questions, suggestions and bug-fixes.

[9]Personal communication, January 15, 2022.